Journal of Digital Imaging

# Building a Open Source Framework for Virtual Medical Training

Ana Cláudia Melo Tiessi Gomes de Oliveira[1] and Fátima de Lourdes dos Santos Nunes[2]

**This paper presents a framework to build medical training applications by using virtual reality and a tool that helps the class instantiation of this framework. The main purpose is to make easier the building of virtual reality applications in the medical training area, considering systems to simulate biopsy exams and make available deformation, collision detection, and stereoscopy functionalities. The instantiation of the classes allows quick implementation of the tools for such a purpose, thus reducing errors and offering low cost due to the use of open source tools. Using the instantiation tool, the process of building applications is fast and easy. Therefore, computer programmers can obtain an initial application and adapt it to their needs. This tool allows the user to include, delete, and edit parameters in the functionalities chosen as well as storing these parameters for future use. In order to verify the efficiency of the framework, some case studies are presented.**

**KEY WORDS: 3D simulation, collision detection, deformation, framework, open source, stereoscopy, virtual reality, 3D imaging (imaging, three-dimensional), cancer detection, computer assisted detection, computer graphics, computer simulation, computers in medicine, education, medical, Graphical User Interface (GUI, human-computer interaction, imaging, three-dimensional, informatics training, medical informatics, applications, programming Languages, teaching, medical training**

## INTRODUCTION

Virtual reality (VR) is a very interesting technology in the development of tools for medical training because it offers interaction in a virtual environment (VE) using mouse and keyboard or non-conventional devices, such as gloves, head mounted displays, joysticks, and haptic devices with force feedback.[1] VR can provide visualizing the details of the procedure in a VE or in a virtual patient. Additionally, it can give feedback to the surgeon about his performance. Computer-generated three-dimensional (3D) environments provide a more advanced human–computer interaction and allow navigation, selection, and manipulation of objects in the virtual world as well as in the real world.[2]

Computing and communication technologies are widely used in the medical area, and VR has attracted interest, since it raises the possibilities of studies and practice of several techniques and medical procedures.[1] The use of VR in medical applications can ensure substituting test objects (like guinea pigs and corpses) for objects modeled to simulate human organs and tissues in the VE. Simulation-based training using VR techniques is a promising alternative to the traditional training of minimally invasive surgeries (MIS). Simulators let the trainee touch, feel, and manipulate virtual tissues and organs using a virtual tool similar to the tool used in actual MIS. The VE allows verifying interactions between the virtual tool and virtual tissues on a monitor as in real laparoscopic procedures.[3]

VR applications in medicine must have special characteristics and requirements, as the use of haptic devices, development of 3D objects to simulate human organs and equipments used in the procedures, collision detection, stereoscopy, and deformation techniques, in addition to real-time

[1]*From the Centro Estadual de Educação Tecnológica Paula Souza/FATEC Garça, 2331 Presidente Vargas Av. Garça, São Paulo, 17400-000, Brazil.*

[2]*From the Escola de Artes, Ciências e Humanidades (EACH), Universidade de São Paulo, São Paulo, Brazil.*

*Correspondence to: Ana Cláudia Melo Tiessi Gomes de Oliveira, Centro Estadual de Educação Tecnológica Paula Souza/FATEC Garça, 2331 Presidente Vargas Av. Garça, São Paulo, 17400-000, Brazil; tel: +55-14-3471-4723; fax: +55-14-3471-4723; e-mail: anatiessi@gmail.com*



*Online publication 30 September 2009*
*doi: 10.1007/s10278-009-9243-3*

feedback and accuracy.[4] In order to make the applications more realistic, it is necessary to plan the VE to reproduce real procedure aspects and actions considering these mentioned aspects. To avoid the implementation of these aspects in every built application, the idea of the source code reuse can be an advantage.[4]

There are VR applications for teaching, rehabilitation, medical procedures training, and simulation. Good examples provided by the literature are simulators for training in minimally invasive surgery,[3] orthopedic simulator,[5] prostate cancer diagnosis,[6] breast plastic surgery planning,[7] and telerehabilitation.[8]

An object-oriented framework (OOF) is a reuse technique with the object-oriented paradigm. In this context, it can describe a framework as a structure of classes that makes available and originates a non-finished application. These classes can generate a set of applications of a previously established domain. Besides the reusability in the analysis, project, programming, and tests stages, an easier maintenance and a rise in productivity are the advantages of using frameworks to develop applications.[9]

This paper presents the development of the virtual medical training (ViMeT), an OOF that uses VR to simulate medical training, considering biopsy exams as the case study. It also presents the ViMeTWizard, a tool to assist the ViMeT instantiation, and the ViMeT integration with other projects. It demonstrates an experience report of the development process and the instantiation of the framework, thereby contributing to researchers who wish to increase the productivity in the building process of the computer applications in this area.

Besides this introduction, this paper has the following sections: "OBJECT-ORIENTED FRAMEWORKS AND VR FRAMEWORKS" shows the concept of the OOF, some of their classifications, and some examples of VR frameworks. "THE VIMET FRAMEWORK" describes the ViMeT's project, the methodology used in its development, and the ViMeTWizard tool. "CASE STUDIES AND DISCUSSION" shows some case studies resulting from the framework instantiation, including a discussion about them. "INTEGRATION OF THE VIMET WITH OTHER PROJECTS" shows the ViMeT's integration with a module of interaction and a system for simulating 3D objects, while the last section presents the conclusions and future works.

## OBJECT-ORIENTED FRAMEWORKS AND VR FRAMEWORKS

The development of OOF and applications derived from a specific framework has been a research subject since the 1980s, with the rise of the object-oriented paradigm. Reusability through framework occurs by inheritance and polymorphism, and as in every software reuse, it presents some limitations. In order to overcome these limitations, it is possible to use simultaneously some forms of reuse such as frameworks with design patterns[10] or frameworks with components.

The way in which the reuse occurs through framework generates a classification according to its extension, and three categories are proposed: black-box reuse, which occurs by composition; white-box reuse, which occurs by inheritance; and gray-box reuse, which has concrete and abstract classes, occurring by inheritance and dynamic linking.[9]

In the last years, an increasing interest in OOF has been observed due to its flexibility and extensibility. The building of a framework can save time in the development of new applications, as well as provide significant reduction of errors and, consequently, increase software quality. The main desired characteristic when developing a framework is the generalization related to concepts and functionalities of a previously established domain.[9]

VR deals with computer-generated 3D environments and a set of tools that allows the user to immerse, navigate, and interact with objects in the VE.[11] For this interaction, conventional devices are used, such as keyboard and mouse, or more complex ones, such as data gloves, haptic device, and stereoscopic glasses. They provide a more realistic interaction in some applications.

VR frameworks are known for their complexity, instantiation difficulty, and for being used only by researchers or expert programmers. They allow the user to focus on the application development, since it is not necessary to worry about the VR system management. Such frameworks offer device and projection system abstraction, specific Scene

Graphs, heuristic interaction with the VE, support to distributed systems, and distributed rendering.[1]

Some examples of VR frameworks are the following: the Avango, which allows the creation of applications with specific classes that inherit properties of distribution;[12] shares simple virtual environment (SSVE), projected to manage small collaborative groups inside highly dynamic, shared and interactive VEs;[13] Ivory, which provides easy visualization of Physics-based data;[14] basho, used to create a VE which supports different renderings and also has a small shell;[15] and simulation open framework architecture (SOFA), which consists of a structure of classes designed to achieve medical simulation in real time.[16]

## THE VIMET FRAMEWORK

As stated before, VR can be helpful to the medical area with regard to applications of computer-aided diagnosis, procedures simulation, 3D visualization, and teaching and training of students and professionals of the medical field. Some researchers previously conducted implementation results in useful functionalities to the medical training such as accurate collision detection, stereoscopy, and deformation.[1] However, it is well known that there are a large variety of techniques of deformation, collision detection, and stereoscopy. Each one can be more or less appropriate to VR applications in medicine. The ViMeT was built in order to consider this context, taking advantage of the better characteristics of the techniques, providing a flexible structure of classes to build applications. In this way, ViMeT consists of an OOF that applies VR techniques considering the ViMeT as a domain and focusing especially on the development of applications that simulate biopsy exams.[4]

The ViMeT framework has computer programmers as its target audience, but the applications generated are from professionals from the health care area (mainly physicians and students) as their target audience. In order to assure the code and functionality access, it is currently being developed under the General Public License,[17] and the full package will be available in http://each.uspnet.usp.br/lapis.

A biopsy exam is a procedure meant to extract small samples of organ tissue suspected of anomalies. These parts are sent to pathologists to determine the diagnosis. The simulation of such exams requires common tasks related to manipulation of equipment and of the human body parts, as well as tasks related to the necessary feedback for simulating the reaction of an action by the user, such as to deform any given area of the synthetic object.

Open source technologies in the ViMeT development was used because it implies that several researchers can contribute to its continuity. The technologies used in ViMeT were Java Programming Language[18] and the API Java3D,[19] database manager system (DBMS) Derby,[20] an educational version of 3D StudioMax for modeling,[21] and Jude[22] for documentation.

### Methodology

Some methodologies are cited in the literature for the development of frameworks. In order to contextualize the methods, the most used are presented here. The methodology proposed by Schmidt[23] affirms that the most important thing is not to initiate the project of a *framework* trying to modulate its variability and flexibility. Instead, a fixed application should be designed inside the frameworks' domain, and after full comprehension, a generalization can be initialized.

Pree[24] proposes a methodology in which the building begins with a definition of the specific object model of an application, and the other activities are successively repeated until the *framework* is considered satisfactory. In this methodology, a developer and a specialist are required isn the domain of the application and both will be co-responsible for the whole process.

Roberts and Johnson's methodology[25] is called "Evolution of Frameworks" and is related to building the instantiation. The general rule is to build the first application, build the second one quickly and different from the first one, and finally to build the third application differently from the others, but all of them must be inside the same domain; thus, the common abstractions will remain evident.

Bosch[26] proposes a methodology in which the building initiates with a domain analysis and six activities. This last methodology was chosen as the defined steps for the building (Fig. 1) were considered the most suitable to the interests of the project. Some adjustments were performed,

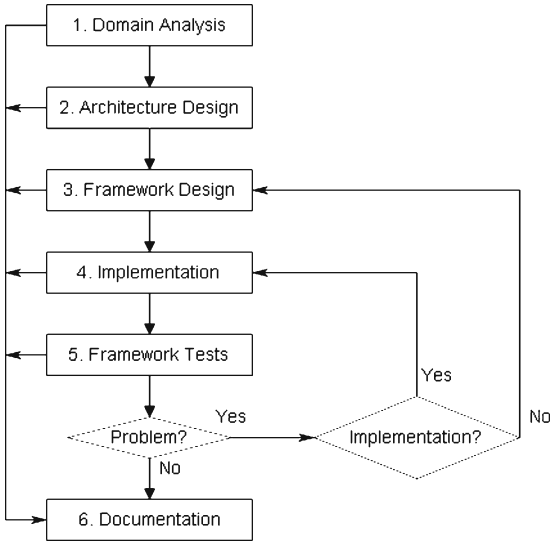


**Fig. 1. Methodology Flowchart of ViMeT Building.**

including the development of a tool for instantiation. The following paragraphs describe the steps:

(a) Step 1, domain analysis: this step was responsible to identify the needed requirements for the framework's domain. For the ViMeT's development, the domain analysis was divided into two parts: The first one was for obtaining the VR framework characteristics, while the second one was intended for obtaining details of the medical procedures necessary to simulate biopsy exams. Regarding the domain analysis related to the VR frameworks, the following characteristics were investigated: VE, database, ways of interactions, ways of loading objects, programming languages, and ways of extension and flexibility. In the analysis related to the biopsy's procedures, the needed requirements for the exam's execution were studied, such as instruments, parts of the human body or organs, and analysis of the collected material. As a result of this stage, some characteristics that influenced the next phases of the development process were established. It was decided that the applications should contain a dynamic VE with two modeled objects: one for representing the human organ and another for representing the medical instrument. Furthermore, functionalities such as stereoscopy, collision detection, and deformation should be included. The use of a DBMS was necessary to store all data from the objects and the generated applications; therefore, the maintenance of the applications could be performed effortlessly. Other established points were the development of a graphic interface to assist the instantiation and the decision that the class structures should be prepared for a future integration of non-conventional devices, once they were not available in the initial version.

(b) Step 2, architecture design: the framework's architectural project was designed based on the domain analysis model. Its purpose is to map a general overview of ViMeT as shown in Figure 2. This diagram shows the types of ViMeT instantiation, which can be performed directly from ViMeT or through the ViMeTWizard tool and the databases that store the objects modeled and the applications generated by ViMeTWizard.

(c) Step 3, framework design: the purpose of this stage was the building of the class diagram foreseen for the framework (Fig. 3). This is a preliminary design projected to consider the insertion of new classes, which corresponds to new functionalities that can be developed to increase the framework in the future. The starting point of this phase was an analysis of the source codes, documentation, and interface of previously developed applications. It was

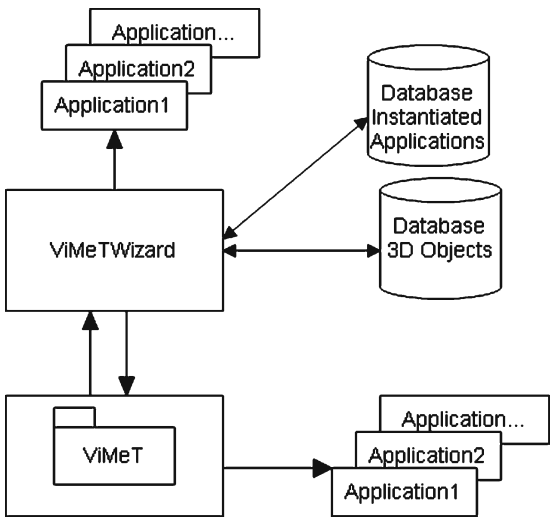


**Fig. 2. ViMeT's Architectural Project.**

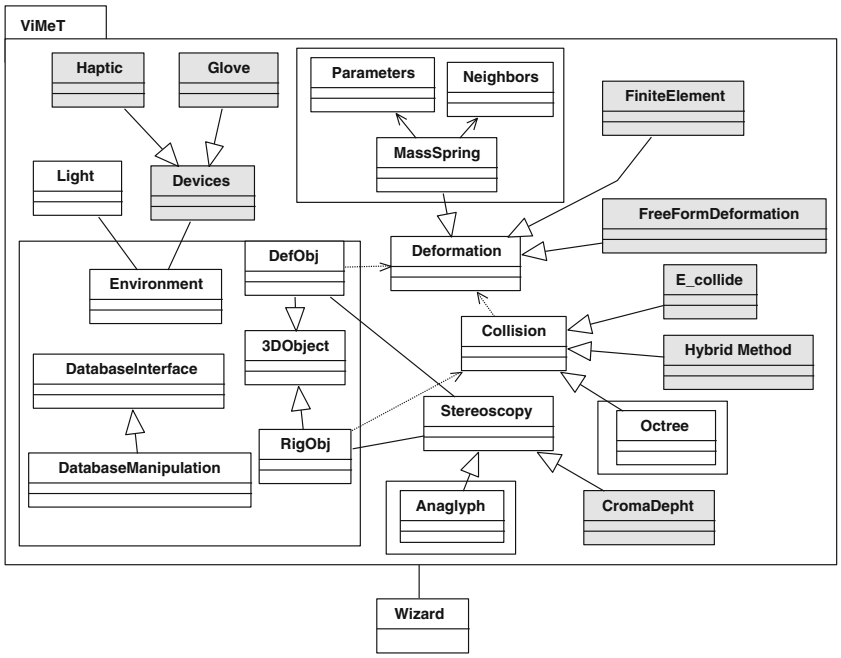


**Fig. 3. ViMeT's class diagram.**[4]

foreseen the reuse of three developed applications (deformation, collision detection, and stereoscopy), each one with an interface and its own methods to import objects (Fig. 4a, b, e and c). Each application is detailed as follows. The deformation appears in VR applications in order to provide greater realism to the environments that contain flexible objects. The change in objects is necessary as an answer to the interaction of the user. The ideal procedure is that the deformation occurs in real time by replacing the object vertices considering new positions and achieved by mathematic equations.[1] Meanwhile, ViMeT provides the mass-spring technique, based on laws of Physics, which allows the reformulation of flexible objects using the concept of mass nodes connected by springs.[27] Detecting a collision is to verify the moment when there is a sufficiently small approximation among the objects in a VE in order to cause a superimposition among them. An application previously developed[1,4]—and reused in ViMeT projects—provides a procedure to detect collisions from the refining of the BoundingBox and BoundingSphere (BS) methods used by Java3D API.[19] Stereoscopy is related to the ability of seeing in three dimensions, i.e., to notice depth.[1] The anaglyph technique was performed in a reused application in the ViMeT project.[4] This technique enables depth information using a pair of images in different colors (red and blue) that correspond to two perspectives of a same object visualized by the same-colored lenses of a pair of glasses, whose function is to filter the images. In this stage of the development process, a comprehensive analysis of the source code of each application was conducted, and it was found that some parts of the source code should be generalized in order to be used in the framework. However, other parts could not be altered because they represented the particularities of each application.

(d) Step 4, implementation: this step was conducted considering the architectural design (Fig. 2), the class diagram (Fig. 3), and the three previously developed applications. This phase was divided into nine activities presented as follow:

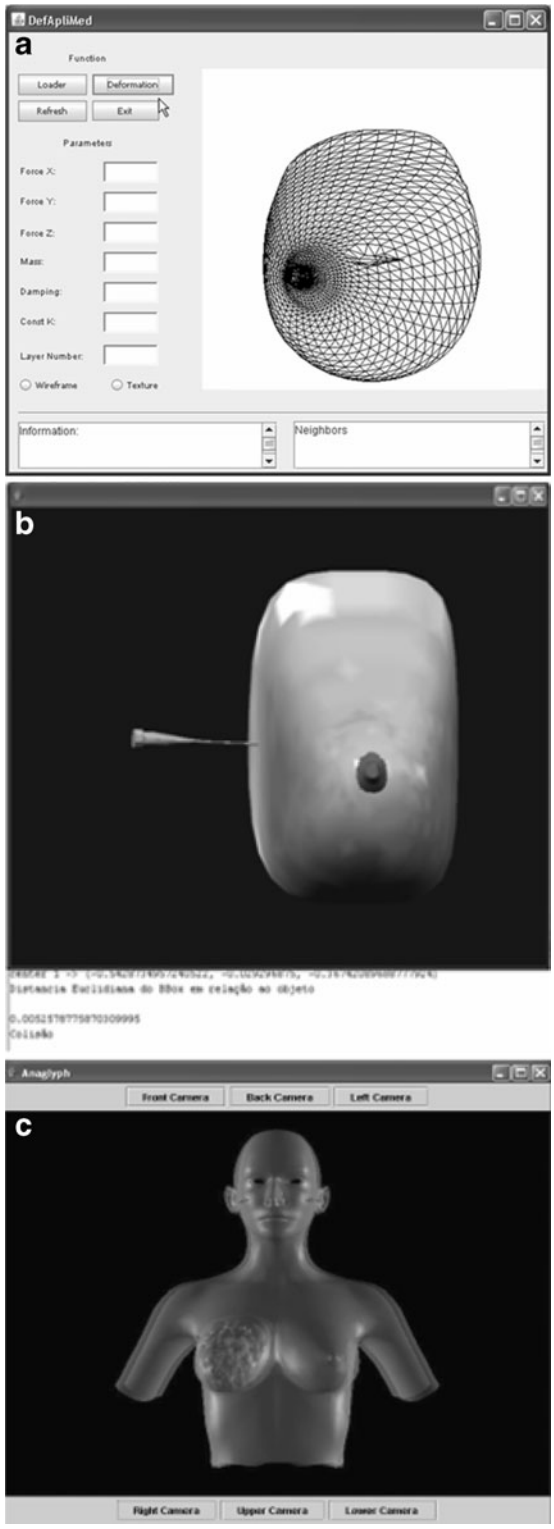


**Fig. 4. Interfaces of the original applications: a deformation, b collision detection, c stereoscopy.**

- Virtual environment creation: the super class VirtualUniverse of the Java3D API [22] was used, which allows the creation of more generic VEs and provides a structure for the insertion of 3D visualization devices and for manipulation. The use of such class provides more flexibility to the VR frameworks implemented in Java because it allows the developer to define the necessary parameters for the VE configuration: objects positioning, 3D visualization, and use of devices for interaction.
- Synthetic 3D objects load: its purpose was to standardize the method of importing 3D objects since there is the need to make the vertices and edges available to some objects to be deformable.
- Integration of the deformation procedure: this activity was performed so the result and performance would be similar to the original application.[4] After extracting the parts of the source code from the original application, tests were carried out in order to verify whether the functionalities in ViMeT presented the same behavior that was observed in the original application.
- Integration of the collision detection procedure: as in the previous activity, some parts of the original application were used after eliminating parts of the source code related to the interface and interaction. Thus, a comparison with the original application was conducted to confirm if its functionality in the framework was similar.
- Integration of the stereoscopy procedure: this implementation dealt with the visualization of objects in the VE. In order to guarantee the ViMeT flexibility, stereoscopy is an optional functionality in the generation of an application.
- Elaboration of the scene graph: this activity was developed simultaneously with the implementation of the aforementioned phases. The scene graph is an organized hierarchic structure of the objects that compose and define a scene, thus preserving the relationships among them. These objects refer to the geometry and its properties and to the necessary information for the rendering of a particular point of view.[21] Figure 5 presents the scene graph for an application with stereoscopy created using ViMeT. The Group nodes are called BranchGroup (BG1 to BG5), and their function is to contain the objects that can be dynamically inserted or removed from the scene in run time. The TransformGroup

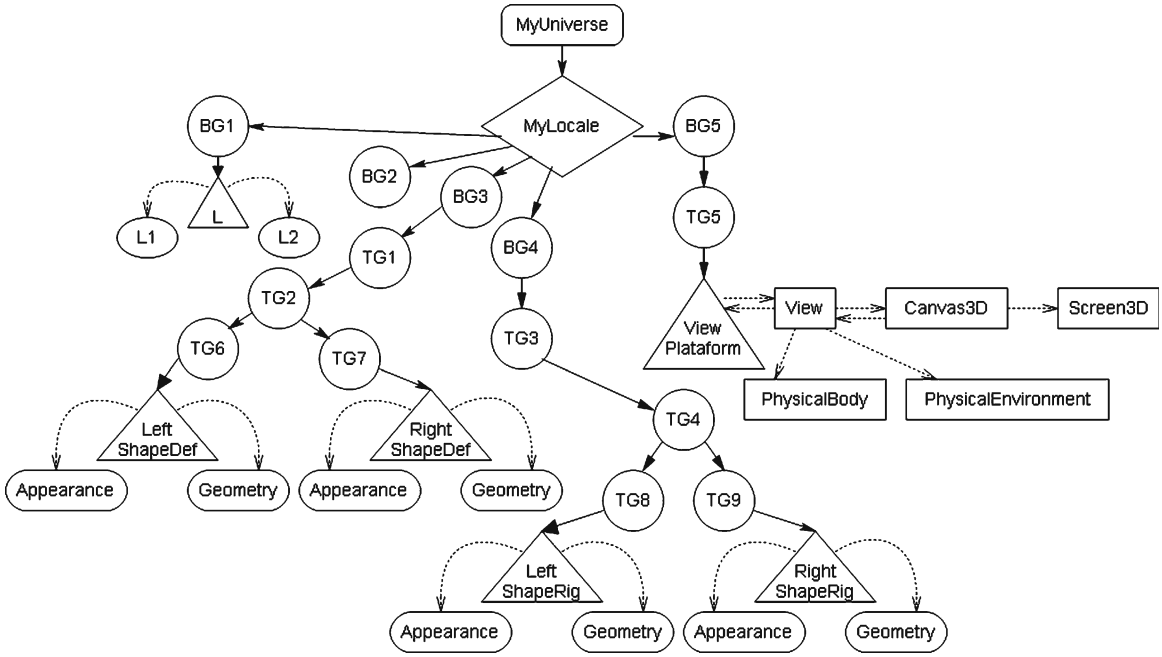


**Fig. 5. Scene Graph generated by ViMeT (application with stereoscopy).**

nodes (TG1 to TG9) are responsible for allowing change in position, orientation, and size of the visual objects in the VE. Every node is associated to the MyLocale node in run time. The function of the MyLocale node is to control the positioning of objects in the VE, and all the other nodes are attached to it. The Leaf nodes (ShapeDef, ShapeRig, *ViewPlataform*, and L) represent scene properties such as geometry and light, while the NodeComponent nodes (L1, L2, appearance, and geometry) are responsible for specifying the properties of an object. The other objects (View, Canvas3D, Screen3D, PhysicalBody, and PhysicalEnvironment) represent classes responsible for visualization, VE location, and user position in the VE, respectively.

- Generalization of parameters: one of the difficulties in implementing classes to a framework is making them sufficiently generic to ensure an efficient instantiation. In ViMeT, the functionalities receive values from parameters that can be registered in a database in order to allow the maintenance of new applications or the generation of new applications from one previously created by using the framework.
- Database integration: this activity involved the development of a database to store all data of the modeled objects and generated application, as mentioned before. The DBMS Derby[19] was used, which is developed in Java; thus, the integration process of the ViMeT becomes easier.
- Instantiation tool building: in this activity, the ViMeTWizard tool was developed simultaneously to the ViMeT development. This tool manages the instantiation of classes and methods, as described in "METHODOLOGY."

(e) Step 5, tests: in parallel to the implementation phase, the testing phase was performed through the direct instantiation of classes (white box) and by the ViMeTWizard tool (gray box). In this step, the tests were conducted in order to prove the development facilities using the Java programming language and API Java 3D. During each activity, new applications were created in order to exercise the implemented code, thus enabling to verify its accordance with the requirements. The goal of the tests conducted was to verify and prove the flexibility of the ViMeT. For that, several

applications were developed, for instance, an application varying the number of functionalities (deformation, collision detection, and stereoscopy), the number of vertices and edges, modeled objects, and the parameters' values (scale, rotation, translation, mass, parallax, and others). The application was executed, hence verifying the computer and visual performance of the generated application. The behavior of each functionality related to the different parameters was also evaluated.

(f) Step 6, documentation: a Cookbook was built in the documentation phase. This is a document to assist in the use of ViMeT and ViMeTWizard, where the main classes and methods are described. Furthermore, the javadoc documentation was developed, a standard used by Sun, which presents every hierarchic structure of super and subclasses. The cookbook consists of a ViMeT installation manual, also explaining details about the configuration of the DBMS Derby's database. In addition, it contains a detailed explanation of how to store a modeled object, alter a generated and stored application, how to use the ViMeTWizard, and develop an application through the ViMeT's class packages. The documentation of the ViMeT allows the programmer to have a clear view of the ViMeT, and therefore, it is possible to use it in the development of several applications. This information is available at http://www.each.usp.br/lapis/vimet.

### The ViMeTWizard Tool

The ViMeTWizard is a tool developed to make the ViMeT instantiation easier. The interface was developed with guides in order to facilitate the visualization of the functionalities and their parameters.

Using the interface, the user can select the objects that represent human organs and medical instruments, define the objects' characteristics related to the scale, translation, and rotation, as well as to define which functionalities will compose the application.

After choosing the characteristics and the objects, the system creates the VE, generates and compiles the source code, thus creating a final application. Every selected characteristic is stored in the database for future changes or review. From the available source code, the user can include certain particular characteristics, hence generating a derived application. Figure 6 shows the development of an application and presents the "Loader" guide, in which the user can choose the objects or load an existing application and then modify parameters, such as scale, translation, and rotation.

The development of the ViMeTWizard demanded the use of a DBMS to store the data of the modeled objects and generated applications. In order to facilitate the instantiation of the ViMeT classes, the DBMS should be simple; thus; the Derby was chosen because it constitutes a simple Java API. Besides, it is an open source software, which allows developing a low-cost application and source code access.

The database has two tables for data persistence. The first one, named OBJECTS, stores data of the 3D modeled objects, which can be representations of human organs (for instance: breast, leg, arm, and others) or medical instruments used in training (needle, syringe, scalpel, among others). The second one, named APPLICATION, is used to store data of the applications generated for later maintenance.

## CASE STUDIES AND DISCUSSION

The expected results of a framework are the reuse of classes in order to generate new applications, simple development, in addition to guaranteeing that the applications comply with the requirements defined in the domain analysis. As aforementioned, there are two ways of using the ViMeT: direct instantiation of the classes and instantiation through the ViMeTWizard. The "white-box" instantiation[9] was chosen for the manual use and the "gray-box"[9] for instantiation using the ViMeTWizard. The manual instantiation uses the ViMeT classes straightforwardly, and the developer can implement new functionalities, as deemed necessary. In the gray-box manner, the instantiation is automatic because the ViMeTWizard executes it. In order to verify the results, five virtual objects were selected: two referring to medical equipment (a needle and a syringe) and three representing human organs (a breast, a leg, and a buttock) as determined in the domain analysis.

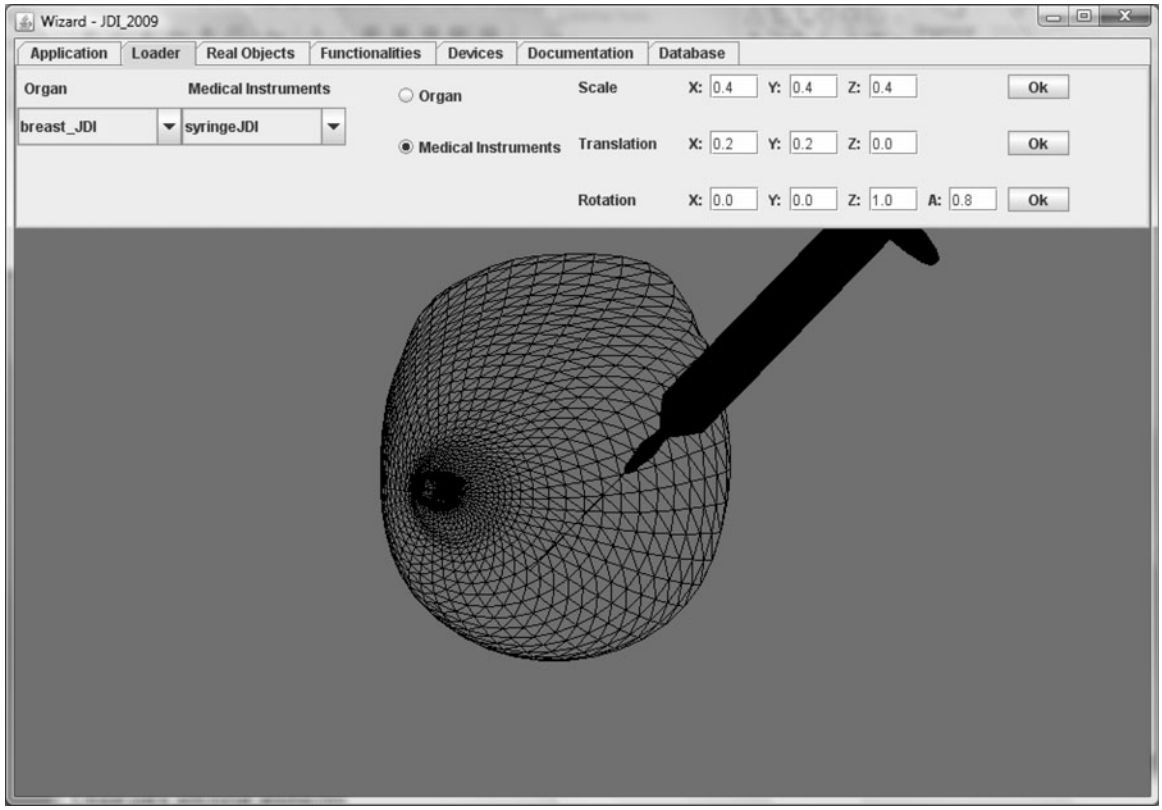


**Fig. 6. ViMeTWizard Interface with an example of an application development.**

Combinations were performed among these objects, with a variation of size and localization parameters in the VE, selection of functionalities, and parameters inherent to the each developed functionality. It is important to point out that this combination and the parameters variation generate different applications.

Figure 7 shows examples of applications generated using the ViMeTWizard tool. Figure 7a presents an application with an object simulating one breast and a thin-needle syringe, including the functionalities of deformation, collision detection, and stereoscopy. Figure 7b shows an application with an object simulating the buttocks and using the same syringe of the previous case, and the functionalities used were deformation and collision detection, while Figure 7c shows an object simulating a leg with a larger number of vertices and edges. The functionalities used were deformation and collision detection.

Figure 8 presents an application generated directly from the manual instantiation of the ViMeT classes; it used an object simulating a breast and another one simulating the medical instrument, considering the functionalities of deformation and collision detection.

Figure 9a exhibits an application generated with the assistance of the ViMeTWizard. Figure 9b shows the source code generated by the tool.

From the ViMeT development and the obtained results, some observations related to the faced difficulties, solutions, and contributions of ViMeT are pointed out. First, it was possible to notice that the methodology used[25] did not foresee the use of finished applications. The difficulties found for this point were regarding the standardization of the loader methods, the interaction and interface of each one of the three reused applications, and the lack of documentation. Therefore, we realized that a previous planning should be conducted when the team intends to reuse source code from previously developed applications. This is necessary because different programmers use different ways of writing and documenting their codes. If a standardization is not established for all applications, substantial effort to adapt the code to the framework is necessary, as was the case for the ViMeT project.

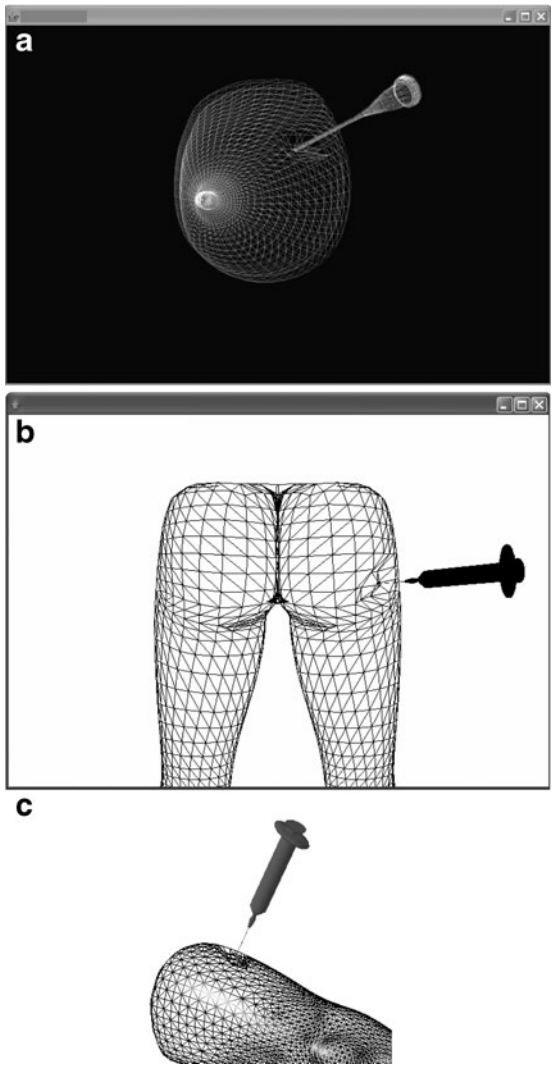


**Fig. 7. a–c: Applications developed through ViMeTWizard (gray box).**

In the VR area, particularly, it is interesting to have a standardization in the definition of the VE and in the building of the scene graph, using the same methods to import and create the interaction with the virtual objects. Regarding the documentation, it is extremely important to build it in a clear and objective manner, without ambiguities, and preferably using tools that assist in the standardization. Using the Java language—the case of the present project—a good standardization is offered when javadoc is used because it is already well established in the users' community of this language.

Another interesting difficulty is the definition of technical questions, like the choice of the best type of VE, the best method to load objects, as well as the best ways for the VE interaction. An in-depth research of these questions and solutions were conducted with the Java language as background. The choice should consider flexibility features and simple integration. Then, we opted for the VirtualUniverse for the VE, Object File for the loader, and Sensor Class for the interaction. With these definitions, the ViMeT provides user flexibility, without which the developer has to worry about these questions related to the basic environment of creation.

The used methodology is very suitable to assist the mentioned standardization process and the correction of errors because it allows for adaptations in its stages. Therefore, it offers the possibility of changing the project during the implementation, but it has a testing stage, parallel to the entire process that enables discovering and correcting errors easily, in the course of any development phase.

Another important issue is that an object orientation allows the development of frameworks in the VR area, making possible faster applications inside a specific domain, hence providing better quality. ViMeT was built in order to act as a standard for the development of new applications. From now on new functionalities must be implemented as part of a bigger project rather than isolated applications. The current results allow future developers to focus only on the implementation of new functionalities once the ViMeT already provides the interface and the VE. Thus, the time spent with the development of future applications can be better used in the specific objectives of each application.

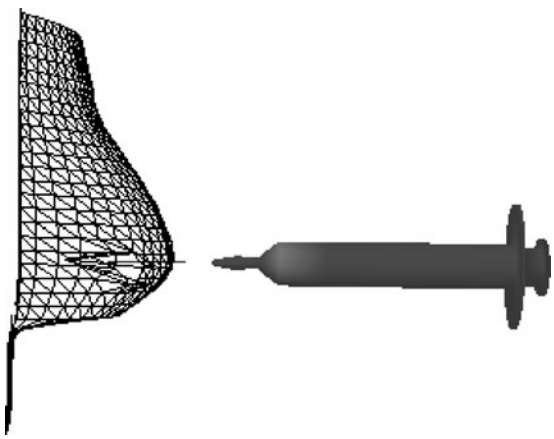

**Fig. 8. Application developed directly from ViMeT's classes (white box).**

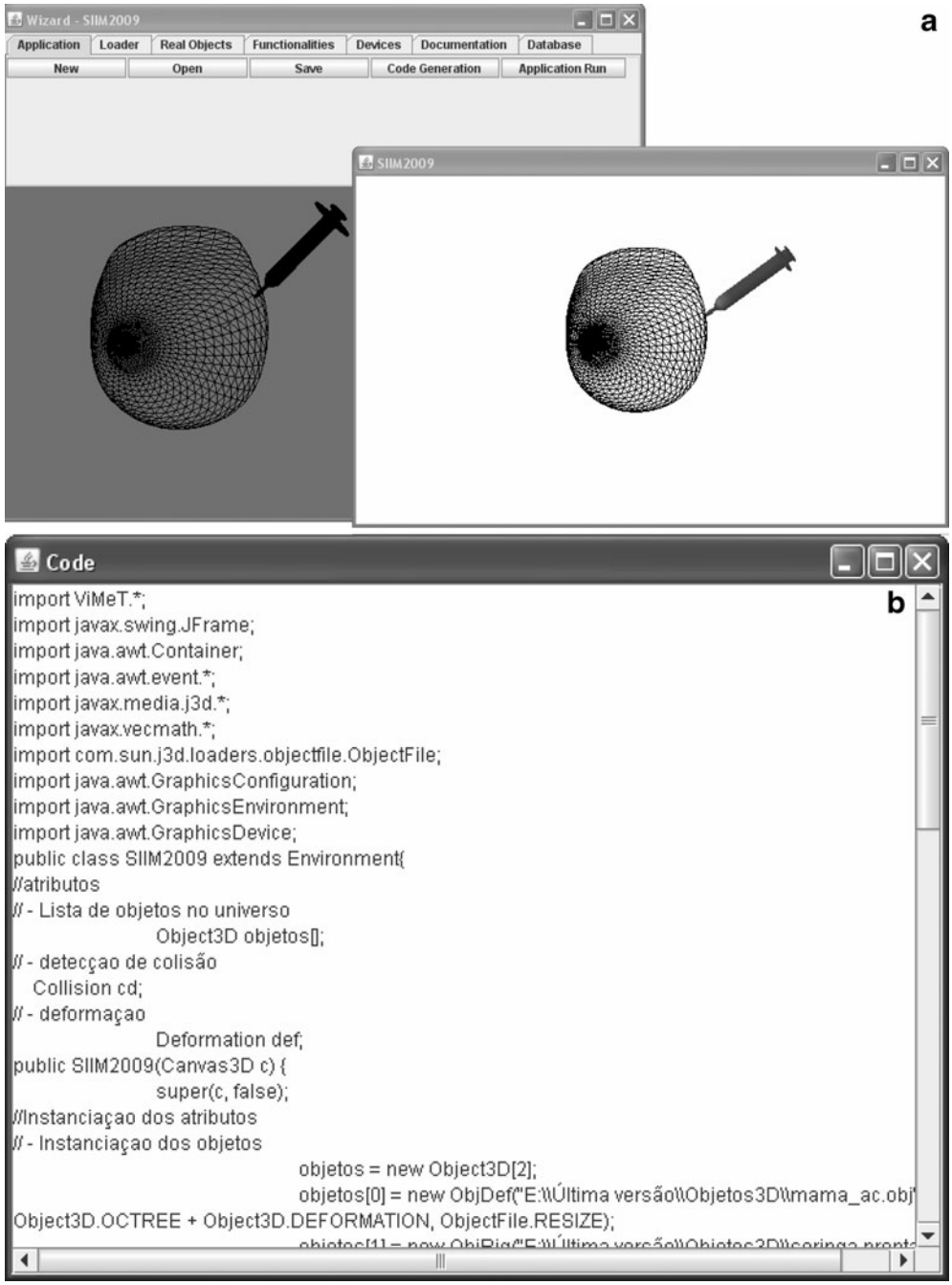


**Fig. 9. a) Example of application development in the ViMeTWizard and application results. b) Source code generated by ViMeTWizard.**

It is important to point out that ViMeTWizard facilitated the ViMeT instantiation for the alterations of the parameters, showing that the manipulation of these parameters in the database is simple and does not imply in performance loss. The generated source code can be adapted by developers accustomed to using the Java programming language to customize new applications, and the application generated through ViMeTWizard can be generated as many times as the user desires, until the expected result is obtained.

In this manner, the main advantages verified for the ViMeT are the simple and quick manner to develop new applications in the framework's domain, the

possibility of manipulating the applications, and the easy integration of new classes representing new functionalities and interaction techniques.

Among the mentioned methodologies, the chosen one contains a testing phase that assists the software's development with quality, in addition to the fact that it seems to be like the standard development of an object-oriented software for documentation.

Finally, it is important to show a comparison of ViMeT's characteristics with other VR frameworks mentioned in "OBJECT-ORIENTED FRAMEWORKS AND VR FRAMEWORKS." Table 1 shows their main features. It is possible understanding that ViMeT is the only one that joins two important characteristics: open source, also including a free DBMS, and multiplatform. The use of Java language makes the application intrinsically multiplatform, so the programmer does not need to know which is the execution environment. It is important to stress that none of them uses an automatic instantiation tool, except for ViMeT. On the other hand, this comparison highlights some aspects of the ViMeT that must be improved in the future: offering other free BDs and extension of domain.

## INTEGRATION OF THE VIMET WITH OTHER PROJECTS

The ViMeT framework, as previously mentioned, is not a finished project, in view of the fact that it is related to a research field that requires high degrees of realism and accuracy, ViMeT is integrated with other projects: (a) simulation of 3D objects, (b) interaction module, and (c) evaluation by doctors and medical students.

In relation to the first one, in parallel to the building of the ViMeT, a 3D simulation system using image processing and VR techniques were developed aiming at extracting the shape and measures from 2D images from a mammography.[28] The second project is related to a system that includes classes in the ViMeT to allow interaction by using dataglove and haptic devices. The third project is for the evaluation of generated applications, which will be detailed as follow.

After the interaction module integration, the generated applications were evaluated by health professionals and a computer professional (volunteers for the tests) through training simulations using conventional (mouse and keyboard) and non-conventional (dataglove and haptics) devices. Examples of simulations using non-conventional equipments are shown in Figures 10 and 11. The task requested to the users was to manipulate the virtual medical instrument until a collision with the virtual human organ was detected by the system. A list of questions was used to collect information [29] about the applications, devices, and sensations that the users noticed in the simulations.

In this evaluation, objects considering two ways of texture were available (Fig. 12). The first model used texture composed by colors, and the second one used a file in JPEG format added to the

**Table 1. Comparison between ViMeT and other VR Frameworks**

| Frameworks | Compared features | | | | | | |
|---|---|---|---|---|---|---|---|
| | Way the reuse | Were design patterns used? | Were Components used? | Programming language | DBMS | Platform | Domain |
| Avango | White-box | No | No | C++ e Scheme | No | Linux and IRIX | Distributed applications |
| SSVE | White-box | No | No | C++ | No | Windows, IRIX, Linux, and Solaris | Dynamic and collaborative groups |
| *IVORY* | White-box | No | No | Java | SQL, DB2 e Dbase | Multiplatform | Visualization of Physics-based data |
| Basho | White-box | No | No | C++ | No | Linux and Mac OSX | VE which supports different renderings |
| SOFA | White-box | No | No | C++ | No | Windows and Mac OS | Medical simulation in real time |
| ViMeT | White-box and Gray-box | No | Yes | Java and API Java 3D | Dervy | Multiplatform | Biopsy exams simulation |

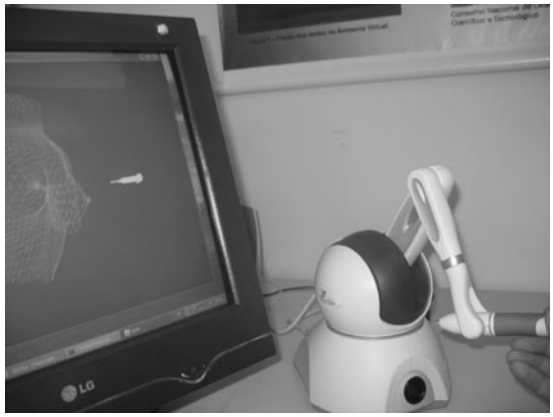

Fig. 10. Example of simulation using haptic device.

wireframe object. However, the users considered that the objects with textures presented more difficulty to manipulate and to notice a sense of depth. Therefore, for the following tests, it was decided to maintain only the wireframe objects. A research about more suitable textures is currently being conducted, and this question will be improved in future versions of ViMeT.

As described by Correa et al.,[28] the volunteers had difficulties mainly with the mouse and haptic device because of the type of task requested. The suggestions given by the volunteers were very important to define ways for improving the ViMeT and for the building of other frameworks of this nature or medical applications. One of the main changes brought about from this evaluation is the modeling of objects. Users considered that a more complex object representing the human body could be better to provide the spatial localization. As a result, new objects were built, as shown in Figure 13. These objects consider other layers besides the surface used before. As can be observed, they include inner structures like a nodule and breast ducts. The volunteers believe that VR systems can help in the medical training that is used in several procedures.

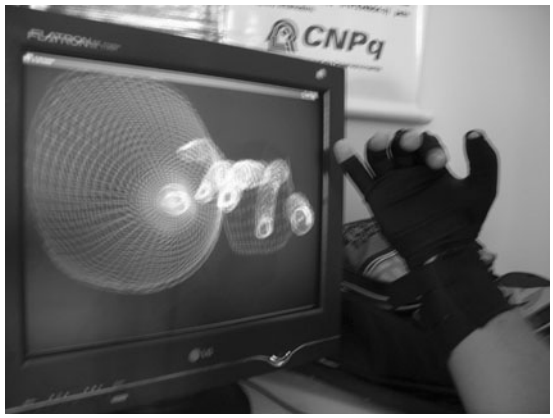


Fig. 11. Example of simulation using dataglove.

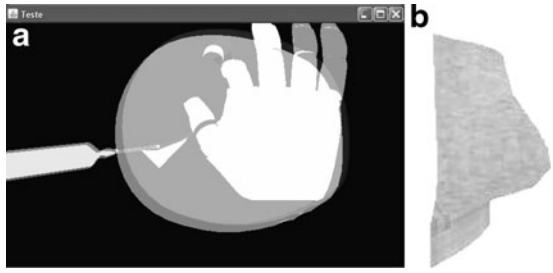


Fig. 12. Examples of objects with textures: a) Texture using colors, b) Texture using an image in JPEG format.

## CONCLUSION AND FUTURE WORK

This paper presented the development process of an OOF for medical training using VR techniques, including deformation, collision detection, stereoscopy, and VE dynamic. One of the requirements during the ViMeT development was a structure of classes that allowed the integration of new functionalities and other interaction devices. Such purpose was achieved with the help of the objected orientation paradigm using Java programming language and the Java3D API. Thus, it was confirmed that the use of a framework could avoid the dependency of a single developer. The Java 3D API also allowed the reduction of the number of classes because it enabled the building of a hierarchic structure of classes composed by superclasses, subclasses, and methods that facilitated the implementation of VEs considering a number of already foreseen functionalities.

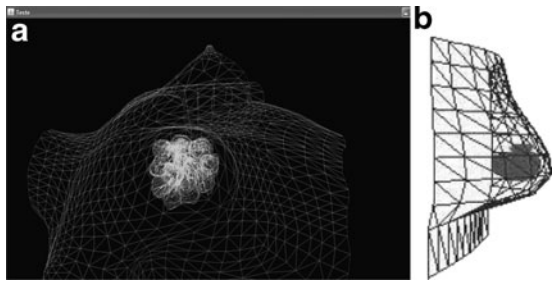


Fig. 13. Example of the more complex objects: a) breast withducts and others tissues; b) breast with nodule.

It was noticed that, in addition to the concern of building an OOF, there is a great concern with regards to the classes hierarchy imposed by the scene graph structure, which contributed to the generalization of parameters, to transforming the objects which became dynamic, as well as the addition and removal of the objects in the VE.

The applications can be generated using the ViMeTWizard innumerable times until the expected result is achieved. Another question was about the performance of the applications generated through ViMeT; however, it was verified that the performance is not altered by the amount of functionalities but by the structures of the modeled objects.

It is important to highlight that, although only biopsy exams case studies were considered, ViMeT classes can be reused in the development of medical training applications in several domains. By the direct instantiation, it is possible to use them in VR applications that belong to other domains and, thus, expand the use of the presented framework.

According to the results obtained in case studies, it can be verified that ViMeT's performance is highly satisfactory and that every result expected with its implementation was achieved. Initial demonstrations for medical professionals show that the tools have full potential to be included in medical teaching in a near future. Nevertheless, it was verified that the generated applications are at a beginning stage and cannot be applied to the practice of medical teaching for a while. In this version, only wireframe objects were included in order to verify the correction of the implemented functionalities. Nowadays, new objects considering textures and the internal composition of the human organs are under study. The deformation process has also been modified in order to consider the full composition of objects. Meanwhile, this question does not involve the implementation or change in ViMeT but only the acquisition or modeling of more suitable objects. The 3D simulation mentioned in "INTEGRATION OF THE VIMET WITH OTHER PROJECTS" is an initial effort to solve this question. Thus, more realistic applications can be obtained after the implementation of the current studies.

Although the ViMeT framework has some aspects to be improved, the generated applications showed that the use of VR application can be promising to become cheaper, motivating, and more effective as medical training for students. In addition, the procedure can be repeated countless times without injuring patients until the student acquires the desired ability.

## ACKNOWLEDGMENTS

We would like to thank the financial support of CNPq.

## REFERENCES

1. Oliveira ACMTG, Pavarini L, Nunes FLS, Botega LC, Justo DR, Bezerra A: Virtual reality framework for medical training: implementation of a deformation class using Java. In: ACM SIGGRAPH international conference on virtual-reality continuum and its applications in industry, 2006, Hong Kong. Proceedings, Nova York: ACM Press, 2006, pp 347–351

2. Liu A, Tendick F, Cleary K, Kaufmann C: A survey of surgical simulation: application, technology and education, vol. 12. Cambridge: MIT Press, 2003

3. Basdogan C, Sedef M, Harders M, Wesarg S: VR-based simulators for training in minimally invasive surgery. IEEE Comput Graph Appl 27(2):54–66, 2007

4. Nunes FLS, Oliveira ACMTG, Rossato DR, Machado MIC: ViMeTWizard: Uma ferramenta para instanciação de um framework de Realidade Virtual para treinamento médico. In: XXXIII Conferencia Latinoamericana de Informática, 2007, San José. Proceedings , 2007, v. 1, p. 1–8

5. Sourin A: Virtual orthopedic surgery training. IEEE Comput Graph Appl 20:6–9, 2000

6. Burdea G, Patounakis G, Popescu V, Weiss RE: Virtual reality-based training for diagnosis of prostate cancer. IEEE Trans Biomed Eng 46(10):1253–1260, 1999

7. Balaniuk R, Costa I, Melo J: Cosmetic bresat surgery simulation. VIII symposium on virtual reality, pp. 387–396

8. Costa R, Carvalho L: The acceptance of virtual reality devices for cognitive rehabilitation: a report of positive results with schizophrenia. Comp Methods Prog Biomed 73(3):173–182

9. Fayad M, Johnson R, Schmidt D: Building application frameworks: object-oriented foundation of frameworks design, Nova Iorque: Wiley, 1999

10. Gamma E, Helm R, Vlissides RJR: Design patterns: elements of reusable object-oriented softwareReading: Addison-Wesley, 1995

11. Greenleaf W: Medical Applications of Virtual Reality. Available at http://www.greenleafmed.com/publications/VR%20Med%20overview.pdf. Accessed Nov. 2005, 2004

12. Tramberend H: Avango: A Distributed Virtual *Reality Framework*. In: Proceedings, Afrigraph '01, ACM, 2001

13. Linebarger JM, Janneck CD, Kessler GD: Shared simple virtual environment: an object-oriented *framework* for highly-interactive group collaboration. In: Proceedings, 7th IEEE DS-RT Conference, 2003. pp 170–180. Available at http://www.cse.lehigh.edu/~dkessler/Publications/LinebargerDS-RT2003.pdf. Accessed 9 Sept 2005

14. Sprenger TC, Gross M, Bielser D, Strasser T: IVORY - An Object-Oriented *Framework* for Physics-Based Information Visualization in Java In: Proceedings of the IEEE Symposium

on Information Visualization (InfoViz'98), IEEE CS Press, 1998, pp 79–86

15. Hinkenjann A, Mannub F: basho—a virtual environment *framework*. In: Proceedings VII Symposium on Virtual Reality. São Paulo: SBC–Brazilian Computer Society, 2004, pp 344–346

16. Allard J, Cotin S, Faure F, Bensoussan PJ, Poyer F, Duriez C, Delingette H: SOFA—an open source framework for medical simulation. In: Proceedings, MMVR, 2007.

17. Sun. The JavaTM Tutorial. Available at http://java.sun.com/docs/books/tutorial. Accessed June 2006

18. Sun (2007) Java 3D API Tutorial. Circulation electronic pages: available at http://java.Sun.com/developer/onlineTraining/java3D

19. Apache. Apache Derby Tutorial. Available at db.apache.org/derby/papers/DerbyTut/index.html

20. 3D Studio Ma: Autodesk 3ds Max, Home page. Available at http://www4.discreet.com/3dsmax, Accessed Nov. 2006.

21. Jude. UML Modeling Tool. Available at http://jude.change-vision.com/jude-web/product/community.html. Accessed set. 2008

22. Pree W: Building application frameworks: object-oriented foundations of *framework* design, New York: Wiley, pp. 379–393, 1996

23. Schmidt HA: Framework design by systematic generalization. In: Fayad M, Johnson R, Schmidt D. Building application frameworks: object-oriented foundation of frameworks design. New York: Wiley, 1999, pp. 353–378

24. Roberts D, Johnson R (1996) Evolving frameworks: a pattern language for developing object-oriented Frameworks. In: Proceedings of Pattern Languages of Programs, Illinois, 1996.

25. Bosch J, Molim P, Mattsson M, Bengtsson P, Fayad ME (1999) Building application frameworks: object-oriented foundations of framework design. New York: Wiley, pp. 55–82.

26. Choi KS, Sun H, Heng PA, Cheng JCY (2002) Scalable Force propagation approach for the web-based deformable simulation on soft tissues. In: Proceedings, Web3D'02 ACM, p.185–193, Fevereiro 2002.

27. de Oliveira ACMTG, Delfino SR, Nunes FLS: Integration of a virtual reality framework with a 3D simulation system. In: 16th Medicine Meets Virtual Reality, 2007, Long Beach, California. Proceedings of 16th Medicine Meets Virtual Reality, 2007

28. Correa CG, Nunes FLS, Bezerra A: Evaluation of VR medical training applications under the focus of professionals of the health area. In: ACM-SAC Symposium on Applied Computer, 2009, Honolulu. Proceedings of ACM-SAC 2009 Symposium on Applied Computer, 2009